ECS 189
WEB PROGRAMMING

5/10

Note on using your own computer

- Some people prefer to develop entirely on their own machines
- See "Setting up a Web server" in Interactive Data Visualization for the Web
- This is a wonderful book, btw, and free online!
- Use the port number we assigned you, even on your own machine
- Please make sure your code runs on the "real server" before turning it in; that is where we grade.

PhotoIndex

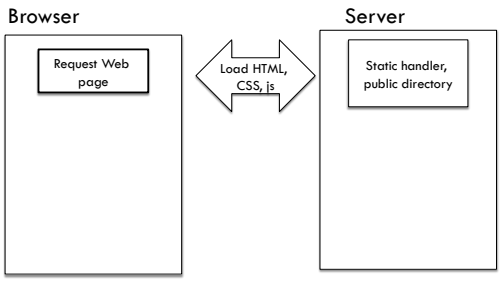
- Upload photos to server
- Get Google Cloud Vision API to suggest what is in the images, producing keywords
- Build database of keywords and images
- Let user browse images by keywords
- Let user delete, correct and add keywords

Google Cloud Vision API Demo

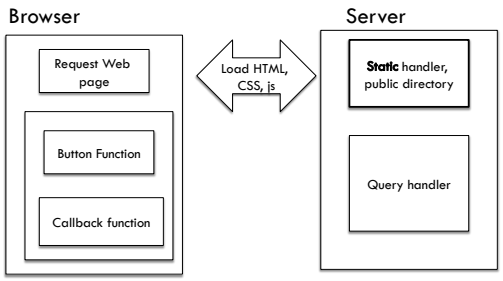
AJAX requests

- Old-school design would send a new Web page every time a query needed to be answered
- Newer Web programming style sends data, often JSON, and then the browser code updates only features of the DOM that need to change.
- Advantages: calmer interface, much of Web page stays the same, no flashing, better user experience
- Asynchronous JavaScript And XML (but often it's JSON instead of XML)

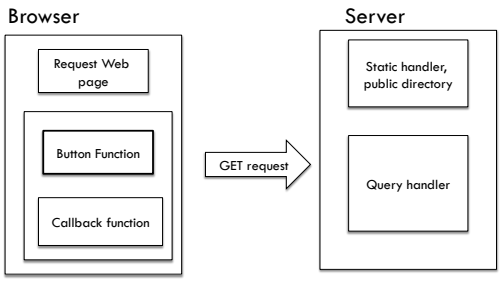
Flow of code



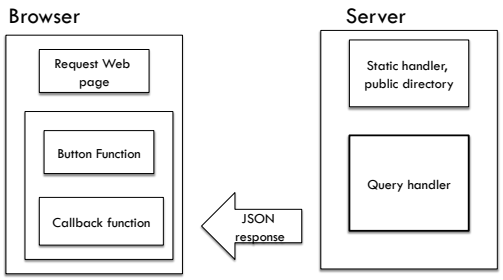
Flow of code



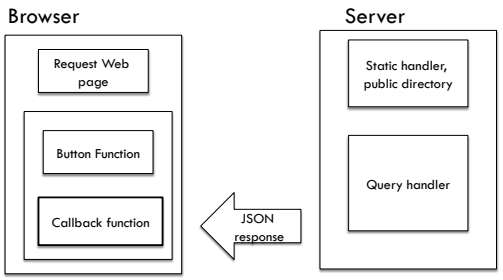
Flow of code



Flow of code



Flow of code



What should the request look like?

- It's a URL with a query:
  138.68.25.50:????/query?....
- We get to make up the query keys and values
- For now, let's make up a query to return labels associated with an image:
  img=hula

On the server

- This is very simple, it will be replaced by something a lot more complicated in our final app
- Hardcode the labels as strings

```
var labels = {hula:
  "Dance, Performing Arts, Sports, Entertainment,
  Quinceañera, Event, Hula, Folk Dance",
  eagle: "Bird, Beak, Bird Of Prey, Eagle, Vertebrate,
  Bald Eagle, Fauna, Accipitriformes, Wing",
  redwoods: "Habitat, Vegetation, Natural
  Environment, Woodland, Tree, Forest, Green,
  Ecosystem, Rainforest, Old Growth Forest"};
```

On the server

- Parse the query and look up the image name

```
if (query) {
    kvpair = query.split("=");
    labelStr = labels[kvpair[1]];
    if (labelStr) {
      response.writeHead(200, {"Content-Type":
"text/json"});
      response.write(labelStr); }
```

Error message for bad query

```
else {
  response.writeHead(404, {"Content-Type": "text/
  plain"});
  response.write("404 Not Found\n");      }

response.end();
```

- response.end() sends either answer.

Works great from browser

http://138.68.25.50:????/hello.html?img=hula

- But how would we get this data from inside a Javascript program?
- Example Web page: labelPix.html; click on image to get labels
- Where in Javascript will we want to send the AJAX request?

AJAX request

- Sent from image's onclick function
- This code is run by the browser, when the button is pushed

AJAX vs JSONp

```
var oReq = new XMLHttpRequest();
```

- When interacting with the Yahoo server, we got data by asking it to download a script.
- Interacting with our own server, we can ask for data directly.
- We do this with an XMLHttpRequest object, which has a bunch of methods to construct and send an HTTP request to the server

Set up URL with query

```
var url = "http://138.68.25.50:60401/query?
  img="+imgName;
```

- As usual, we make the query by pasting together the right URL
- imgName here should be the name of one of the images

Set up a callback

```
function reqListener () {
    var pgh = document.getElementById("labels");
    pgh.textContent = this.responseText;
}
```

- Added to request object as a method, so this refers to the request
- When does this get run?

Send off the request

```
// setup callback
oReq.addEventListener("load", reqListener);
// load occurs when operation is completed,
// response is back.
oReq.open("GET", url);  // writes HTTP req head
oReq.send();  // initiates transfer
```

- This is a GET HTTP request.

Kinds of HTTP requests

- All HTTP requests initiate an exchange with the server. There is no way for the server to initiate an exchange with the browser!
- GET – retrieves data or sends small amount of information in URL. Body is usually empty. Used to retrieve static pages or for queries.
- POST – send data to server, in body of HTTP request.
- There are others but they are rarely used.

XMLHttpRequest

- Can be used for any kind of HTTP request
- Has all the basic parts of a request that we saw before in JSONp
  - URL containing a query string
  - Callback function to handle server response
  - Response shows up in responseText property
- Many frameworks cover XMLHttpRequest up to make it prettier
- There is a JQuery version, a D3 version, etc.
- All basically are this under the hood